**SSN College of Engineering, Kalavakkam Department of Computer Science and Engineering III Semester - CSE 'A ',’B’ & ‘C’**  
**UCS 1312 Data Structures Lab**

Aditya Krishna P

185001010

CSE A

**Exercise 12: Implementation of Dijkstra algorithm to find shortest Path from the given source vertex to all the other vertices of the graph**

(i)  Represent the graphs using adjacency matrix or adjacency list.

(ii)  Implement Dijkstra algorithm to find shortest path from source vertex to all other vertices (Source: ‘V1’ - Graph 1 and ‘1’ - Graph 2)

#include <stdio.h>

#include <stdlib.h>

#define notavertex -1

#define V1 7

#define V2 6

#define MAX 10

#define false 0

#define true 1

#define infinity 1000

typedef int vertex;

typedef struct tableentry {

int known;

int dist;

vertex path;

} tableentry;

typedef tableentry table [V1];

vertex choosevertex (table t, int x) {

int size;

if(x==1) size=V1;

else size=V2;

vertex min;

for (int i=0; i<size; i++)

if(t[i].known==false)

min=i;

for (int i=0; i<size; i++)

if(t[i].known==false && t[i].dist < t[min].dist)

min=i;

return min;

}

void inittable (vertex start, table t, int x)

{ int size;

if(x==1) size=V1;

else size=V2;

for (int i=0; i<size; i++) {

t[i].known= false;

t[i].dist = infinity;

t[i].path = notavertex;

}

t[start].dist = 0;

t[start].path = 0;

}

int loopcondition (table t, int x) {

int size;

if(x==1) size=V1;

else size=V2;

for (int i=0; i<size; i++)

if(t[i].known == false)

return 1;

return 0;

}

void displaytable (table t, int x) {

int size;

if(x==1) size=V1;

else size=V2;

printf("V K D P\n");

for (int i=0; i<size; i++) {

printf("%d %d %5d ", i+1, t[i].known, t[i].dist);

if(t[i].path!=notavertex && i!=0)

printf("%d\n", t[i].path+1);

else printf("%d\n", t[i].path); }

printf("--------------\n\n");

}

void djikstra (table t, int graph [][V1], int x ) {

vertex v,w;

int size;

if (x==1)

size=V1;

else

size=V2;

while (loopcondition(t,x)) {

v=choosevertex(t,x);

t[v].known = true;

for (w=0; w<size; w++) {

if(graph[v][w] != 0)

if(t[w].known == false)

if (t[v].dist + graph[v][w] < t[w].dist) {

t[w].dist = t[v].dist + graph[v][w];

t[w].path = v;

}

}}}

void printpaths (int start, int end, table t, int path[], int \*size, int \*dist) {

if (end != start) {

path[(\*size)++] = t[end].path +1;

\*dist += t[end].dist;

printpaths (start, t[end].path, t, path, size, dist);

}

}

void reversepath (int path[], int size) {

for (int i=size-1; i>=0; i--)

printf ("%d ", path[i]);

}

int main () {

printf("---GRAPH 1---\n");

int graph1[V1][V1] = { {0,2,0,1,0,0,0}, {0,0,0,3,10,0,0}, {4,0,0,0,0,5,0}, {0,0,2,0,2,8,4}, {0,0,0,0,0,0,6}, {0,0,0,0,0,0,0}, {0,0,0,0,0,1,0} };

table t1;

inittable(0, t1,1);

djikstra(t1, graph1,1);

displaytable(t1,1);

for (int i=1; i<V1; i++) {

printf("Path from V1 to V%d:\n", i+1);

int path[MAX];

int size=0, dist=0;

printpaths(0,i,t1, path, &size, &dist);

reversepath(path, size);

printf("%d", i+1);

printf("\nDISTANCE: %d\n", dist);

printf("\n"); }

printf("\n---GRAPH 2---\n");

int graph2[V1][V1] = { {0,5,0,6,10,0,0}, {5,0,1,0,2,7,0}, {0,1,0,0,0,8,0}, {6,0,0,0,3,0,0}, {10,2,0,3,0,4,0}, {0,7,8,0,4,0,0}, {0,0,0,0,0,0,0} };

table t2;

inittable(0,t2,2);

djikstra(t2, graph2, 2);

displaytable(t2,2);

for (int i=1; i<V2; i++) {

printf("Path from V1 to V%d:\n", i+1);

int path[MAX];

int size=0, dist=0;

printpaths(0,i,t2, path, &size, &dist);

reversepath(path, size);

printf("%d", i+1);

printf("\nDISTANCE: %d\n", dist);

printf("\n"); }

return 0;

}

**OUTPUT**

gml17:Ex12 csea17$ gcc Da.c

---GRAPH 1---

V K D P

1 1 0 0

2 1 2 1

3 1 3 4

4 1 1 1

5 1 3 4

6 1 6 7

7 1 5 4

--------------

Path from V1 to V2:

1 2

DISTANCE: 2

Path from V1 to V3:

1 4 3

DISTANCE: 4

Path from V1 to V4:

1 4

DISTANCE: 1

Path from V1 to V5:

1 4 5

DISTANCE: 4

Path from V1 to V6:

1 4 7 6

DISTANCE: 12

Path from V1 to V7:

1 4 7

DISTANCE: 6

---GRAPH 2---

V K D P

1 1 0 0

2 1 5 1

3 1 6 2

4 1 6 1

5 1 7 2

6 1 11 5

--------------

Path from V1 to V2:

1 2

DISTANCE: 5

Path from V1 to V3:

1 2 3

DISTANCE: 11

Path from V1 to V4:

1 4

DISTANCE: 6

Path from V1 to V5:

1 2 5

DISTANCE: 12

Path from V1 to V6:

1 2 5 6

DISTANCE: 23